Advance 86a Personal Computer.

Preliminary manual.

chkdsk : 131072 - 128 x 1024
102704

28368

Introduction.

Welcome to the Advance 86a personal computer.

This manual contains everything you need to know to unpack, setup your system and begin running programs. The Advance can display and process information in a wide vareity of different forms - as text, text and graphics or a mixture of both. With its sixteen bit microprocessor, large memory, and powerful interfaces the Advance can handle many tasks that have been beyond the "home" or "low-cost" computers that have been available up until now. When you add the Advance system expansion unit the Advance becomes a disk based system that is hardware and software compatible with the IBM Personal computer - giving you instant access to the largest base of software available for any personal computer.

The Advance system unit is the centre piece of your system. Inside there is an 8086 microprocessor, RAM and ROM memory, and the necessary electronics to allow the Advance to communicate with you via a television or monitor display, a keyboard, and loudspeaker. Other connectors allow a cassette recorder, parallel printer, joysticks, light pen and expansion units to be plugged in.

Input to the system unit is through an 84 key keyboard which includes an alphanumeric section, separate numeric pad, and ten programmable function keys.

System setup

Read carefully through the entire setup documentation BEFORE attempting to switch on your Advance computer. Treat the keyboard and system unit carefully - they are sophisticated pieces of electronics.

NOTE:

> Always remember. Before plugging or unplugging any cables, moving the system, or installing peripherals, make sure THE COMPUTER IS SWITCHED OFF AND DISCONNECTED FROM THE MAINS SUPPLY.

System setup consists of:

1. Unpacking.
2. Acquainting yourself with the various pieces of equipment that will be connected together.
3. Connecting the cables that link the various elements together.
4. Switching on for the first time.

Equipment needed:

1. The Advance 86a system unit.
2. The Advance 86 keyboard unit.

3. Mains power cord.
4. A domestic television or video monitor.
5. A video cable suitable for the type of display you are using.
6. An audio cassette recorder.
7. A cassette lead suitable for the cassette recorder.

If you are unsure about any of the above equipment your supplier will be able to help.

Clear a reasonable space on your work area. You will need easy access to both the front and rear of the Advance 86a system unit.

Position the system unit so that the plexiglass front is facing you.

Place the keyboard in front of the system unit. The plug on the coiled keyboard cable fits into the socket on the front of the system unit under the power switch. Make sure the plug is firmly seated - it should mate easily and there is no need to use any degree of force.

Check that the power switch is in the OFF (UP) position.

First connect the power cord to the socket at the back of the system unit and then to a mains outlet.

Position the power switch to ON. After a second or two you should hear one short beep from the loudspeaker, indicating that the system unit has completed its self-test.

If the power switch does not light, switch off, disconnect the power cord from the mains outlet, and check the power cord, fuse and plug.

If the loudspeaker gives a series of long or short beeps then the system unit is faulty. Contact your supplier.

SWITCH OFF AND DISCONNECT THE POWER CORD FROM THE MAINS OUTLET. YOUR COMPUTER, VIDEO DISPLAY, CASSETTE RECORDER SHOULD BE SWITCHED OFF AND DISCONNECTED FROM THE MAINS AT THIS TIME.

The next step is to connect your video display to the computer.

Connecting a television

Connect the video cable to the output of the UHF modulator at the back of the Advance syustem unit. Connect the other end of the video cable to the aerial input of your television.

Plug the system unit and televisicn into the mains supply. As before after a second or two the computer should issue one short beep. Using the tuning control on your television you should be able to get a stable, legible display on the screen.

The display may be visible at several different positions of your TV's tuning control. You can use whichever setting produces the best display.

Connecting a video monitor

For a composite video monitor connect the video cable to the phono connector at the rear of the system unit. If you are using an RGB monitor plug the video cable into the DB9 connector at the rear of the system unit. Connect the other end of the video cable to the input connector on the video monitor. Plug the system unit and video monitor into the mains outlet and switch on. The computer should again issue a short beep, indicating that it is working correctly. On the video monitor you should see the Advance Basic sign on message.

If you are unable to get a satisfactory display on your television or monitor contact your supplier for assistance.

Connecting a cassette recorder

Connect your cassette lead to the DIN style connector at the rear of the system unit. Connect the other end of your cassette lead to the cassette recorder.

From Advance Basic the statements that control the cassette recorder are:

```
LOAD
SAVE
BLOAD
BSAVE
MOTOR
```

Cassette filenames should be enclosed in quotes.

With your Advance there should be a sample cassette with a program named colour on it. You can use this program to adjust your video display to get a high quality display.

To load colour, type

```
LOAD "COLOUR",R
```

The program will run immediately after it has loaded. If you get an error message whilst the system is loading adjust the volume and tone controls on your cassette recorder and try again.

Advance Basic

Advance Basic may be used in either of two modes: direct mode or indirect mode. In direct mode, statements and commands are not preceded by line numbers. They are executed as they are entered. Results of arithmetic and logical operations may be displayed immediately and stored for later use, but the instructions themselves are lost after execution. Direct mode is useful for debugging and for using the Advance Basic interpreter as a calculator for quick computations that do not require a complete program.

Indirect mode is used for entering programs. Program lines are preceded by line numbers and stored in memory. At any time the program stored in memory can by executed by entering the RUN command.

Line format

Advance Basic program lines have the following format:

⟨line number⟩ ⟨Basic statement⟩[:Basic statement...] ⟨carriage return⟩

More than one Advance Basic statement may be placed on a line, but each must be separated from the last by a colon.

An Advance Basic program line always begins with a line number and ends with a carriage return. Line numbers indicate the order in which the program lines are stored in memory. Line numbers are also used as references in branching and editing. Line numbers must be in the range 0 to 65529. A line may contain a maximum of 255 characters.

A full stop character (.) may be used in EDIT, LIST, AUTO, and DELETE commands to refer to the current line.

Active and visual display pages.

Every command that writes to or reads from the screen transfers information to the active screen. Advance Basic lets you switch between different active screens using the SCREEN statement. The screen which is actually being displayed is called the visual screen.

Character set

The Advance character set consists of alphabetic characters, numeric characters, and special characters. In all a total of 256 different characters can be displayed. In the graphics modes you can redefine the character set at will.

Constants

Constants are the values Advance Basic uses during execution. There are two types of constants: string and numeric.

A string constant is a sequence of up to 255 alphanumeric characters enclosed in double quotation marks.

For example:

```
"Hello world"
"$25,000"
"Hi there"
```

Numeric constants are positive or negative numbers. Advance Basic numeric constants cannnot contain commas. There are five types of numeric constants:

1. Integer constants — Whole numbers between -32768 and 32767. Integer constants do not contain decimal points.

2. Fixed-point constants — Positive or negative real numbers.

3. Floating-point constants — Positive or negative numbers represented in exponential form.

4. Hex constants — Hexadecimal numbers, denoted by the prefix &H.
   For example:
   &H76

5. Octal constants — Octal numbers, denoted by the prefix &O or &.
   For example:
   &0377

6. Binary constants — Binary numbers denoted by the prefix &B.
   For example:
   &B011

Single/double precision form for numeric constants

Numeric constants may be either single precision or double precision numbers. Single precision numeric constants are stored with 7 digits of precision, and printed with up to 6 digits. Double precision numeric constants are stored with 16 digits of precision and printed with up to 16 digits.

A single precision constant is any numeric constant that has one of the following characteristics:

1. Seven or fewer digits.

2. Exponential form using E.

3. A trailing exclamation point !

Examples:

```
46.8
-1.09E-06
3489.0
22.5!
```

A double precision constant is any numeric constant that has one of these characteristics:

1. Eight or more digits

2. Exponential form using D.

3. A trailing number sign (£).

Examples:

345692811
-1.09432D-06
3489.0£
7654321.1234

Variables

Variables are names used to represent values used in an Advance Basic program, the value of a variable may be assigned explicitly by the programmer, or it may be assigned as the result of calculations in the program. Before a variable is assigned a value, its value is assumed to be zero (or the null string "" for a string variable).

Variable names and declaration characters

Advance Basic variable names may be of any length. Up to 40 characters are significant. Variable names can contain letters, numbers, and the decimal point. However, the first character must be a letter. Special type declaration characters (listed below) are also allowed.

A variable name may not be a reserved word, but embedded reserved words are allowed. Reserved words include all the Advance Basic commands, statements, function names, and operator names. If a variable begins with FN, it is assumed to be a call to a user defined function.

Variables may represent either a numeric value or a string. String variable names can be written with a dollar sign($) as the last character. For example A$="sales report". The dollar sign is a variable type declaration character, that is, it "declares" that the variable will represent a string.

Numeric variable names may declare integer, single precision, or double precision values. The type declaration characters for these variable names are as follows:

% Integer variable

! Single precision variable

£ Double precision variable

The default type for a numeric variable name is single precision.

Variable types may also be declared by including the Advance Basic statements DEFINT, DEFSTR, DEFSNG, and DEFDBL in a program.

Array variables

An array is a group or table of values referenced by the same variable name. Each element in an array is referenced by an array variable that is subscripted with an integer or an integer expression. An array variable name has as many subscripts as there are dimensions in the array. For example V(10) would reference a value in a one-dimensional array, T(1,4) would reference a value in a two-dimensional array, and so on. The maximum number of dimensions for an array is 255. The maximum number of elements per dimension is 32767.

Variable space requirements

Type	Bytes
Integer	2
Single precision	4
Double precision	8
Integer array	2 per element
Single precision	4 per element
Double precision	8 per element

Strings can be up to 255 characters in length.

Keyword entry

Advance Basic allows you to enter common Basic keywords quickly using the Alt key and a single alphabetic key (A-Z). The table below shows which keys are assigned keywords.

A - AUTO	N - NEXT
B - BSAVE	O - OPEN
C - COLOR	P - PRINT
D - DELETE	Q - Q
E - ELSE	R - RUN
F - FOR	S - SCREEN
G - GOTO	T - THEN
H - HEX$	U - USING
I - INPUT	V - VAL
J - J	W - WIDTH
K - KEY	X - XOR
L - LOCATE	Y - Y
M - MOTOR	Z - Z

Advance cassette Basic commands, statements, functions.

ABS(x) - returns absolute value of X.
ASC(x$)- returns ASCII code of first letter of string X$.
ATN(x) - returns angle whose tangent is x in radians.
AUTO [<line number> [,<increment>]] - generates line numbers automatically.
BEEP - Sounds loudspeaker at 800Hz for 1/4 second.
BLOAD filename[,offset] - loads a memory image from cassette into memory at offset.
BSAVE filename,offset,length - saves a portion of memory to cassette.
CALL <variable name> [(<argument list>)] - call assembly language subroutine.
CDBL(x) - converts x to a double precision number.
CHR$(x) - Returns a string whose one character has the ASCII code x.
CINT(x) - converts x to an integer.
CLEAR [,<memspace>] [,stack space.]] - allocate memory, stack space. Clears variables, closes open files.
CLOSE [[#]<file number>[,[#]<file number>...]] - close open file(s).
CLS - clears screen.
COLOR [foreground] [,background] [,border] - sets colours in text mode.
COLOR [background] [,[palette]] - sets colours in graphic modes.
CONT - continues program execution after a break.
COS(x) - returns cosine of x in radians.
CSNG(x) - converts x to a single precision floating point number.
CSRLIN - returns current line number of cursor.
DATA <list of constants> - Stores numeric and string constants that are accessed by the READ statement.
DEF FN<name> [(<parameter list>)] = <function definition> - define single line function.
DEF SEG [=address] - define segment for subsequent BLOAD,BSAVE,CALL,PEEK,POKE USR statements.
DEFINT <list of letters> - define variables whose names begin with specified letters as integers.
DEFDBL <list of letters> - define variables whose names begin with specified letters as double precision variables.
DEFSTR <list of letters> - define variables whose names begin with specified letters as string variables.
DEFUSR [<digit>] <=integer expression> - define address of USR function.
DELETE [<line number>] [-<line number>] - deletes program lines.
DIM <list of subscripted variables> - declares array dimensions.
EDIT <line number> - displays a line for subsequent editing.
END - terminates program execution, closes all files, and returns to command level.
EOF (<file number>) - Indicates an end-of-file condition.
ERASE <list of array variables> - Deletes specified arrays.
ERR - returns error code for last error encountered in program.
ERL - returns line number of last error encountered in program.
ERROR (x) - Simulates the occurence of a BASIC error.
EXP(x) - Exponential function. Returns e raised to the power x.
FIX(x) - truncates x to an integer.
FOR <variable name> = <start value> TO <end value> [STEP increment]
NEXT [<variable name> [,<variable name>....]] - performs a series of BASIC statements in a loop a specific number of times.
FRE(x) - returns amount of free storage in BASIC workspace.
FRE(x$) - returns amount of free storage in BASIC workspace. Forces string space garbage collection.

GOSUB <line number>
RETURN - transfers execution to and from subroutine at line number.
GOTO <line number> - transfers execution to line number.
HEX$(x) - returns a string that represents the hexadecimal value of x.
IF ... THEN ..... ELSE - Conditional transfer of control.
IF ... GOTO ..... ELSE - Conditional transfer of control.
INKEY$ - read character from keyboard.
INP(x) - read I/O port number x.
INPUT [;] [("prompt string");] <list of variables> - Returns input from keyboard during program execution.
INPUT £ <file number>,<variable list> - input data from file.
INPUT$ (len[,[£]filenumber]) - inputs string of specified length from file.
INSTR([I,]x$,y$) - searches for string y$ in x$ beginning at position I.
INT(x) - returns integer value of x.
KEY key number, x$ - sets function key to specified string.
KEY LIST - displays current function key assignments.
KEY ON - display function keys at bottom of screen.
KEY OFF - turn function key display off.
LEFT$(x$,I) - returns first I characters of string x$.
LET <variable> = <expression> - assign expression value to variable.
LINE [(x1,y1)] - (x2,y2) [,[colour] [,B[f]]] - draws lines or boxes.
LINE INPUT [;]["prompt string"];] stringvar - input line of text from keyboard
LINE INPUT£filenum,stringvar - input line of text from file.
LIST [linenumber] [-linenumber] [,filename] - lists program text
LLIST [linenumber] [-linenumber] - lists program text on printer
LOAD [filename][,R] - load program from cassette
LOCATE [row] [,[col] [,[cursor] [,[start] [,stop]]]] - position, modify cursor
LOG(x) - returns the natural logarithm of x.
LPOS(x) - returns current position of printhead in buffer.
LPRINT <list of expressions> - prints values of expressions on printer.
LPRINT USING <string expression> ; <list of expressions> - prints formatted values of expressions on printer.
MERGE <filename> - Merges lines from ASCII program file with program in memory
MID$ (x$,I[,I1])
MID$(x$,I[,I1]) = y$ - extract or assign substring.
MOTOR <state> - switch cassette motor on or off.
NEW - deletes program in memory and clears all variables.
OCT$(x) - returns string which is octal value of x.
ON ERROR GOTO <line number> - trap errors.
ON <expression> GOTO <list of line numbers> - Branches depending on value of expression.
ON <expression> GOSUB <list of line numbers> calls a particular subroutine depending on the value of the expression.
OPEN <mode>,[£]<file number>,<filename> [,<record length>]
OPEN <filespec> [FOR <mode>] AS [£]<file number> [LEN=<record length>] - Open data file.
OPTION BASE <0 or 1> - sets base value for array subscripts.
OUT <port number>,<x> - outputs x to specified port.
PEEK(x) - returns the byte pointed to by address x.
PEN ON - Enables light pen read.
PEN OFF- Disables reading of the light pen.
PEN(x)- Read light pen status.
POINT (x,y) - Returns colour of specified point on screen.
POKE x,y - Writes byte y into memory location x.

POS(x) - Returns current column of cursor.
PRESET [(xcoordinate),,ycoordinate)) [,(colour)] - draw point on screen.
PRINT (list of expressions) - print data on screen.
PRINT USING (string exp);(list of expressions) - print formatted information.
PRINT£ (filenumber) (list of expressions) - print information to file.
PRINT£ (filenumber) USING (string) (list of expressions) - print formatted information to file.
PSET ((xcoordinate),(ycoordinate)[,colour)] - plot point
RANDOMIZE[(expression)] - seed random number generator.
READ (list of variables) - read values from data statement.
REM (text) - program comment.
RENUM [[(new number)] [,(old number)] [,(increment)]]] - renumber lines.
RESTORE [(line number)] - Allows DATA to be reread from a specified line.
RESUME (line number) ! (NEXT) ! (0) - continue program execution after error.
RETURN - return from subroutine.
RIGHT$ (x$,I) - return right I characters of sring x$.
RND(x) - return pseudo random number.
RUN [(line number)] - begin program execution.
SAVE filename [,A] or [,P]
SCREEN [mode] [,[burst] [,[apage] [,vpage]]]- alter screen mode.
SGN(x) - returns the sign of x.
SIN(x) - returns sine of x, where x is an angle in radians.
SOUND frequency,duration - sound generation.
SPACE$(x) - returns a string of x spaces.
SPC(x) - skips x spaces in a PRINT, LPRINT or PRINT £ statement.
SQR(x) - returns the square root of x.
STICK(x) - return joystick coordinates.
STOP - terminates program execution, prints break message and returns to command level.
STR$(x) - returns a string representation of the value of x
STRIG - returns status of the joystick buttons.
STRING$(n,m)
STRING$(n,X$) - returns a string of length x, all of whose characters have the ASCII code m (form 1), or all of whose characters are x$ (form 2)
SWAP (variable name),(variable name) - Exchanges the value of two variables.
TAB(x) - Move the print position to x.
TAN(x) - returns the tangent of x. X is the angle in radians.
TRON - enable tracing of program flow.
TROFF - disable tracing of program flow.
USR [(digit)] [(argumnet))] - call an assembly language routine.
VAL(x$) - returns the numeric value of string x$.
VARPTR ((variable name))
VARPTR (£(file number)) - returns address of variable or file buffer.
WAIT (port number),I[,J] - Loop until port has specific value.
WHILE (expression)
WEND - executes series of statements in loop as long as expression is true.
WIDTH [LPRINT] (size) 132
WIDTH (file number), (size)
WIDTH (device),(size) - alter line width.
WRITE - [(list of expressions)] - print data to the screen.
WRITE£(file number),(list of expressions) - print data to a file.

Extended key codes

The INKEY$ statement will return a two character string when certain keys are pressed. If the first character returned by INKEY$ is an ASCII NULL - 00, you should examine the second character to determine which key was pressed. Table shows the second character returned and the corresponding key.

Table

Code	Key
3	NULL
15	Back TAB (!<-)
16-25	Alt- Q,W,E,R,T,Y,U,I,O,P
30-38	Alt- A,S,D,F,G,H,J,K,L
44-50	Alt- Z,X,C,V,B,N,M
59-68	F1-F10
71	Home
72	Cursor Up ( )
73	PgUp
75	Cursor left ( )
77	Cursor right ( )
79	End
80	Cursor down ( )
81	PgDn
82	Ins
83	Del
84-93	Shift F1-F10
94-103	Ctrl F1-F10
104-113	Alt F1-F10
114	Ctrl PrtSc
115	Ctrl Cursor left
116	Ctrl Cursor right
117	Ctrl End
118	Ctrl PgDn
119	Ctrl Home
120-131	Alt 1,2,3,4,5,6,7,8,9,0,-,=
132	Ctrl PgUp

Advance Basic error messages.

Normally when Advance Basic detects an error the program is terminated and one of the error messages below displayed. It is possible to trap and test errors in a BASIC program using ON ERROR, ERR, and ERL.

Number	Message
1	Next without FOR
2	Syntax error
3	Return without GOSUB
4	Out of data
5	Illegal function call
6	Overflow
7	Out of memory
8	Undefined line number
9	Subscript out of range
10	Duplicate definition
11	Division by zero
12	Illegal direct
13	Type mismatch
14	Out of string space
15	String too long
16	String formula too complex
17	Can't continue
18	Undefined user function
19	No resume
20	Resume without error
21	Unprintable error
22	Missing operand
23	Line buffer overflow
24	Device timeout
25	Device fault
26	FOR without NEXT
27	Out of paper
28	Unprintable error
29	WHILE without WEND
30	WEND without WHILE
51	Internal error
52	Bad file number
54	Bad file mode
55	File already open
57	Device I/O error
62	Input past end
64	Bad file name
66	Direct statement in file
67	Too many files
68	Device unavailable

Advance ROM BIOS.

The Advance 86 system has a ROM resident basic input/output system or BIOS. This provides device level control over the I/O devices on the system board. The BIOS routines give a programmer a standard, well defined interface to the system hardware. Most of the BIOS is concerned with communications - with the video screen, keyboard, printer, cassette, etc. In addition the BIOS provides various system services such as a real-time clock, equipment check, memory size determine etc. Table 1 summarises the functions available. Whilst the BIOS routines can be called from almost any language running on the Advance they are really intended for the use of experienced programmers who need a high degree of control over the system. Of necessity little error checking is performed by the BIOS routines and if you accidently misuse them it is perfectly possible to "crash" the machine, losing all the work you have in memory.

Using the BIOS

The BIOS consists of a set of subroutines in ROM. Each subroutine can typically carry out several different functions. To chose which functions you load the appropriate parameters into the 8086 registers. To actually call the desired subroutine you issue a software interrupt.

For example, suppose we want to output a character to the parallel printer port. Parallel printer support is done via INT 17H. According to Table 2, INT 17H supports three different functions selected by loading register AH with 0,1 or 2. Function 0 outputs the character in AL to the parallel port. So, in assembly language the code to call the BIOS would look like:

```
 MOV AH,0  ; Select function 0 - print a character
 MOV AL,13 ; Print a carriage return code - ASCII 13
 MOV DX,0  ; Select the first printer attached to the system
 INT 17H   ; Call printer support routine
; (Printer status is returned in AH)
```

As a rule the BIOS routines preserve all registers except AX and the flags. Note that some routines return status values in the other registers.

Table 1

Interrupt number	Function
10H	Video IO
11H	Equipment check
12H	Memory size determine
13H	Diskette IO
14H	RS 232 communications
15H	Audio cassette support
16H	Keyboard support
17H	Parallel printer
1AH	Real-time clock

Table 2

VIDEO IO interrupt 10H

AH=0 set mode
- AL=0 40 x 25 BW Text
- AL=1 40 x 25 Colour Text
- AL=2 80 x 25 BW Text
- AL=3 80 x 25 Colour Text
- AL=4 320 x 200 Colour graphics
- AL=5 320 x 200 BW graphics
- AL=6 640 x 200 BW graphics

AH=1 set cursor type
- CL bits 0-4 End line for cursor
- CH bits 0-4 Start line for cursor

AH=2 set cursor position
- DL column number
- DH row number
- BH page number - 0 for graphics

CURSOR-TYPE

0
1
2
3
4
5
6
7

AH=3 read cursor position
- BH page number - 0 for graphics
  - returns
  - DL column
  - DH row
  - CL bits 0-4 end line for cursor
  - CH bits 0-4 start line for cursor

AH=4 read light pen position
- BH page number - 0 for graphics
  - returns
  - AH light pen switch status 0 = not down/triggered
    1 = valid reading
  - DL column
  - DH row
  - CH raster line 0-199
  - BX pixel column 0-319 or 0-639

AH=5 change display page in text mode
    AL page number 0-7 in 40 col mode, 0-3 in 80 col mode

AH=6 scroll display page up
    AL number of lines, 0 means entire window
    CL column of upper left corner to scroll
    CH row of upper left corner to scroll
    DL column of lower right hand corner to scroll
    DH row of lower right hand corner to scroll
    BH screen attribute to use on blank lines

AH=7 scroll display page down
    AL number of lines, 0 means entire window
    CL column of upper left corner to scroll
    CH row of upper left corner to scroll
    DL column of lower right hand corner to scroll
    DH row of lower right hand corner to scroll
    BH screen attribute to use on blank lines

AH=8 return character/attribute at current cursor position
    BH screen page
    returns
    AL character at cursor position
    AH attributes of character at cursor position

AH=9 print character/attribute at current cursor position
    BH screen page
    CX number of characters to write
    AL ASCII code of character to write
    BL Attributes of character

A AH=10 print character at current cursor position
    BH screen page
    CX number of characters to write
    AL ASCII code of character to write

B AH=11 set colour palette
    BH colour palette
    BL colour value

C AH=12 write dot
    AL colour value - if bit 7=1 screen = screen XOR AL
    CX column number
    DX row number

D AH=13 read dot
    CX column number
    DX row number
    returns
    AL dot read

E AH=14 write character
    AL ASCII code of character to write
    BL Foreground colour in graphics mode
    BH Display page in alpha mode

F.

```
AH=15 return current video state
     returns
     AL current screen mode
     AH number of character columns on screen
     BH current screen page

Equipment check INT 11H

     returns
     AX = equipment attached to system

Memory size determine INT 12H

     returns
     AX = number of 1K blocks of continuous memory

Cassette IO INT 15H

AH=0 Turn cassette motor on

AH=1 Turn cassette motor off

AH=2 Read 256 byte blocks from cassette

     ES:BX pointer to data buffer
     CX number of bytes to read
     returns
     ES:BX pointer to last byte read + 1
     DX number of bytes actually read
     CY 0 if no error occured, 1 if error in reading data
     AH Error code if CY=1
        01 - CRC error
        02 - data transitions were lost
        04 - no data was found

AH=3 write 256 byte blocks to cassette

     ES:BX pointer to data buffer
     CX number of bytes to write
     returns
     ES:BX pointer to last byte written + 1
     CX 0

Keyboard INT 16H

AH=0 read next character from keyboard
     returns
     AL character
     AH scan code

AH=1 Is character available ?
     returns
     Z flag = 1 no code is available
     Z flag = 0 code is available
     if Z flag = 0 AX next character available in buffer
```

```
AH=2 Return current shift status
     returns
     AL = current shift status

Parallel printer INT 17H

AH=0 Print character
     AL character to be printed
     DX printer number 0,1 or 2
     returns
     AH =1 if timeout error

AH=1 Initialise printer port
     DX printer number
     returns
     AH printer status

AH=2 return printer status
     DX printer number
     returns
     AH printer status


Real-time clock INT 1AH

AH=0 Read current clock setting
     returns
     CX High portion of count
     DX Low portion of count
     AL 0 if timer has not passed 24 hours. Non 0 if another day

AH=1 Set current clock
     CX High portion of count
     DX Low portion of count
```

Advance 86 Connector specifications.

Keyboard connector J1.

Pin 1 Keyboard clock signal
Pin 2 Keyboard data
Pin 3 Not used
Pin 4 0v
Pin 5 +5v

Audio cassette connector J2.

Pin 1 Motor control output 1.
Pin 2 0v.
Pin 3 Motor control output 2.
Pin 4 Data In.
Pin 5 Data Out.

Loudspeaker J3.

Pin 1 Speaker drive.
Pin 2 +5v.
Pin 3 Not used

Joysticks J4.

Pins 1,8,9 & 15 +5v.
Pins 4,5 & 12 0v.
Pin 2 Push button input 1. Bit 4.
Pin 7 Push button input 2. Bit 5.
Pin 10 Push button input 3. Bit 6.
Pin 14 Pushbutton input 4. Bit 7.
Pin 3 Games paddle A, X axis input.
Pin 6 Games paddle A, Y axis input.
Pin 11 Games paddle B, X axis input.
Pin 13 Games paddle B, Y axis input.

Centronics parallel printer J5.

Pins 2,3,4,5,6,7,8 & 9. These pins supply the 8 bit parallel data to the printer. The signals are TTL compatible. Data bit 0 is on pin 2 and data bit 7 on pin 9.
Pins 18 & 25 0v.
Pin 1 strobe (active low).
Pin 10 acknowledge (active low).
Pin 11 Busy (active high).
Pin 12 Paper end.
Pin 13 Printer select.
Pin 14 Auto LF after CR (active low).
Pin 15 Error.
Pin 16 Initialise printer.
Pin 17 Select In.

RGB Video out J6.

Pin 1 & 2 0v
Pin 3 RED
Pin 4 GREEN
Pin 5 BLUE
Pin 6 Intensity.
Pin 7 Composite sync.
Pin 8 Horizontal sync.
Pin 9 Vertical sync.

These lines are all TTL compatible.

Composite video out J7.

This phono style connector supplies a composite video output to drive a composite video monitor - either monochrome or colour.

UHF Modulator

The UHF modulated output comes from the phono style connector on the UHF modulator. This is a PAL encoded composite video signal modulated onto an E36 carrier frequency. It is suitable for driving any domestic television - monochrome or colour.

Advance B6a personal computer system specifications

Microprocessor 8086 running at 4.77 Mhz

ROM 64K bytes containing Microsoft GW Basic interpreter
Power on self test and diagnostic routines
ROM BIOS

RAM 128K bytes as standard, with parity checking
An additional 128K can be plugged into the system board

Basic

Industry standard, compatible with IBM PC cassette BASIC
Full screen editor built in
Extensions for graphics, sound, light pen, joysticks all built in

Advance cassette GW BASIC statements

ABS ASC ATN AUTO BEEP BLOAD BSAVE CALL CDBL CHR$ CINT CLEAR
CLOSE CLS COLOR CONT COS CSNG CSRLIN DATA DEF FN DEF SEG
DEFtype DEF USR DELETE DIM EDIT END EOF ERASE ERR ERL ERROR
EXP FIX FOR..NEXT FRE GOSUB..RETURN GOTO HEX$ IF..THEN..ELSE
INKEY$ INP INPUT INPUT£ INPUT$ INSTR INT KEY LEFT$ LEN LET
LINE LINE INPUT LINE INPUT£ LIST LLIST LOAD LOCATE LOG LPOS
LPRINT LPRINT..USING MERGE MID$ MOTOR NEW OCT$ ON..ERROR
ON..GOSUB ON..GOTO OPEN OPTION..BASE OUT PEEK PEN POINT
POKE POS PRINT PRINT..USING PRINT£ PRINT£..USING PSET PRESET
RANDOMIZE READ REM RENUM RESTORE RESUME RETURN RIGHT$ RND
RUN SAVE SCREEN SGN SIN SOUND SPACE$ SPC SQR STICK STOP STR$
STRIG STRING$ SWAP TAB TAN TRON TROFF USR VAL VARPTR WAIT
WHILE..WEND WIDTH WRITE WRITE£

Video

Uses composite video/RGB/ or domestic television
6845 CRT controller.
Text modes Foreground/background colours programmable.
40 x 25
80 x 25
Foreground 1 of 16 colours
Background 1 of 8 colours
Border 1 of 16 colours

Bit mapped Graphics
160 x 100 16 colours
320 x 200 4 colours
Background 1 of 16 colours, 2 Palettes with 3 colours each
640 x 200 2 colours

256 characters including upper/lower case, scientific,
European languages.

Text and graphics can be mixed freely on screen.
In graphics modes character set can be redefined at will by the user

Keyboard

Separate detached unit 84 keys.

Keyboard layout, characteristics under software control and can be redefined at will.
Keys auto repeat when held down
10 Programmable function keys
Separate numeric/cursor control pad
Keyboard is interrupt driven so characters are never lost

Parallel printer

Industry standard Centronics parallel printer port

Audio cassette interface

Supports loading and saving of programs to audio cassette. BASIC programs, data files, machine language programs and pure binary information such as graphics screens can all be loaded/saved.

Speaker

Loudspeaker programmable under software control

Joysticks/ Game paddle

Connections for two analogue joysticks. Four push button inputs

Light pen

Connector for Advance light pen

Co-processor

Provision for 8087 floating point co-processor chip on board.

Expansion

Can be connected to system expansion unit. Adds twin 360K disk drives, MSDOS, extended BASIC, Supercalc 3, Superwriter, RS232 interface, 4 IBM PC compatible bus slots, two true 16 bit slots.

This document describes the sw ch settings and video set up procedure for the Advance 86a system board

1.- Switch Settings

1.1 Switch 1

This switch is used to define the type of display device connected to the system board. If a R.B.G. monitor is being used then the slider of SW1 should be in the position nearest the card edge. If a composite video or a domestic television is being used then the slider of SW1 should be set in the position nearest to I.C.66.
If the switch is set for R.G.B. monitor when using a domestic T.V. then unintensified characters will appear very faint on the display.

1.2 Switch 2

This is a four position D.I.L. switch located near I.C.93. Each of the four switch elements has the following effects.

1.2.1 Switch 2 element 1

This element is used to indicate if an Advance 86b expansion unit is fitted or not. For Advance 86a unit boards (unexpanded) this switch is set to the ON position.
If an expansion chassis is connected then this switch will be set to the OFF position.

1.2.2 Switch 2 element 2

This element is set to the ON position for use with PAL domestic T.V.s and PAL standard composite video monitors.
it is set to the OFF position for use with NTSC domestic T.V.s and NTSC standard composite video monitors.
For R.G.B. monitors the switch should be set to the OFF position.

1.2.3 Switch 2 elements 3 & 4

These two switches are not used. They will both be set to the OFF postion on all boards.

1.3 Switch 3

This switch is positioned near I.C.109. It is used to determine the signal drive levels to the audio cassette.
The slider is set nearest to I.C.109 for audio casette AUX. input. The slider is set nearest to J2 for audio cassette MIC. input.

2.0 Board Adjustments

2.1 Variable Resistor VR1

This potentiometer can be used to set the D.C. level of the video signal into the Astec modulator to achieve a video sync level (negative peak voltage) to 2.6 volts +/- 0.15 volts.

2.2 Variable Capacitor VC1
This variable capacitor adjusts the chromiance bandwidth of the video signal. VC1 is adjusted for maximum chrominance reference burst measured with an oscilloscope at the Astec modulator video input. The oscilloscope should be triggered on the line sync pulses.

2.3 Variable Capacitor VC2
This variable capacitor is used to adjust the frequency of the chrominance oscillator frequency to acheive a stable colour on a domestic T.V.

Connector Pin Assignments

1.1 Connector J1
This is a 0.1 inch pitch male header used to connect to the keyboard.

Pin 1 Keyboard clock signal
Pin 2 Keyboard Data signal
Pin 3 Keyboard Reset (Not used)
Pin 4 0 Volts
Pin 5 +5 Volts

1.2 Connector J2
This is a 5 way 180 degree D.I.N. connector used to connect to the audio cassette.

Pin 1 Motor control output 1. Used with Pin 3 to control the cassette motor on recorders with a remote control facility.
Pin 2 0 Volts
Pin 3 Motor control output 2
Pin 4 Data In. This pin should be connected to the audio output of the cassette recorder.
Pin 5 Data Out. This pin should be connected to the AUX or MIC input of the cassette recorder. (See also notes on Switch 3)

1.3 Connector J3
This is a 0.1 inch pitch male header used to connect an 8 Ohm loudspeaker.

Pin 1 Speaker Drive Signal
Pin 2 +5 Volts (used as a return from the speaker)

1.4 Connector J4

This is a 15 way D Type socket used to connect the games paddles.

Pins 1,8,9 & 15 +5 Volts
Pins 4,5 & 12 0 Volts
Pin 2 Pushbutton Input 1. When this pin is connected to 0 Volts then bit 4 of the games paddle register will be zero.
Pin 7 Pushbutton Input 2. When this pin is connected to 0 Volts then bit 5 of the games paddle register will be zero.
Pin 10 Pushbutton Input 3. When this pin is connected to 0 Volts then bit 6 of the games paddle register will be zero.
Pin 14 Pushbutton Input 4. When this pin is connected to 0 Volts then bit 7 of the games paddle register will be zero.
Pin 3 Games Paddle "A" , X axis input. (The other side of the games potentiomter should be connected to +5 Volts)
Pin 6 Games Paddle "A" , Y axis input.
Pin 11 Games Paddle "B" , X axis input.
Pin 13 Games Paddle "B" , Y axis input.

1.5 Connector J5

This is a 25 Way D Type connector used to connect a parallel printer.

Pins 2,3,4,5,6,7,8 & 9	Data lines .	(Data bit 0 is on pin 2) (Data bit 0 is on pin 9)
Pins 18 & 25	0 Volts	
Pin 1	Strobe Signal	(Active low)
Pin 10	Acknowledge	(Active low)
Pin 11	Busy	(Active high)
Pin 12	Paper End	(Active high-no paper)
Pin 13	SCLT.	(Active high)
Pin 14	Auto FD.XT	(Active low)
Pin 15	Error	(Active low-if error seen)
Pinl6	Initialise	(Active low-normally high)
Pin 17	Select In	(Active low)

1.6 Connector J6
This is a 9 Way D Type socket used to connect a R.B.G. monitor.

Pins 1 & 2	0 Volts.
Pin 3	Red
Pin 4	Green
Pin 5	Blue
Pin 6	Intensity
Pin 7	Comp. sync. Contains vertical & horizontal sync information.
Pin 8	Horizontal sync
Pin 9	Vertical sync

All the above are T.T.L. compatible signals.

1.7 Connector J6
This is a Phono Type socket supplying a composite video monitor output.

1.8 Modulator U.H.F. or V.H.F. output
This is a phono style connector for used to connect a domestic T.V..

1.9 Connector P1 & P2 (ISSUE 2 BOARDS)
These are 6 way 0.156 inch male headers, P1 is near I.C.5 & 6, P2 is near 1.C.2 & 3, used to connect power.

P1

Pins 1 to 5	+5 Volts
Pin 6	+12 Volts

P2

Pins 2 to 6	0 Volts

1.10 Connector P2 (ISSUE 3 BOARDS)
This is a 6 way 0.156 inch male header used to connect power.

Pin 2 & 3	0 Volts
Pin 4 & 5	+5 Volts
Pin 6	+12 Volts

1.11 Connector P3
This is a 0.1 pitch male header used to connect a light pen.

Pin 1	LPen . Light Pen Signal input.
Pin 3	LPen Sw. Indicates if light pen is in contact with the screen.
Pin 4	0 Volts
Pin 5	+5 Volts
Pin 6	+12 Volts

Advance Expansion Unit Connector

Edge Connectors P1 & P2

P1
This is a 40 pin connector intended to connect pin for pin with edge connector P1 on the system board.(Pin 1 to Pin 1 - Pin 40 to Pin 40)

P2
This is a 34 pin connector intended to connect pin for pin with edge connector P2 on the system board.

RS232 Serial Port

This is a 25 Way D Type Connector.

Pin 2	EIA TX DATA
Pin 3	EIA RX DATA
Pin 4	EIA RTS
Pin 5	EIA CLR TO SND
Pin 6	EIA DATA SET RDY
Pin 7	SIG GND
Pin 8	EIA CARRIER DET
Pin 9	+XM ITCL RET
Pin 11	-XM ITCL DATA
Pin 18	+RCV CL DATA
Pin 20	EIA DTR
Pin 22	EIA RI
Pin 25	-RCV CL RET

Power Connector

Pin 1 & 2	+5 Volts
Pin 3	-5 Volts (Used on IBM cards but not by system)
Pin 4	-12 Volts
Pin 5	+12 Volts
Pin 6 & 7	0 Volts

Floppy Disk Controller Board Connections

Edge connector J1 connects pin for pin with edge connector J1 on Disk Drive "A".(Pin 1 to Pin 1, Pin 34 to Pin 34 inc)
Drive "A" connects to Drive "B" as follows

DRIVE "A"		DRIVE "B"	
J1	Pins 1-9	J1	Pins 1-9
J1	Pins 10-16	J1	Pins 16-10
J1	Pins 17-34	J1	Pins 17-34

Note. A termination resistor pack (150 ohms) is required to be fitted in Drive "B" only.

Procedure for setting up the Floppy Disc Controller Cards

Using an oscilloscope monitor the signal at TP2 and adjust VR2 to obtain a time interval of 2.0 µsecs from leading edge to leading edge. This represents a frequency of 500 KHz.

Monitor TP1 with the oscilloscope and using a DOS diskette in drive A attempt to boot the unit by pressing Ctrl, Alt and Del simultaneously or by switching OFF then ON.

Using VR1 set the pulse width of the positive signal at TP1 to 1 µsec.

Note that the pulse width of the positive part of the signal at TP1 must not exceed the width of positive part of the signal at TP2.